Towards a Front End Architecture
-brewster

Requirements:
  Near optimum performance for numeric and word-size codes.
  Good performance for vp ratios of 4 and higher
  C and Lisp performance
  "Operating system" on 1 machine type
  supports BI VAX, and Lispms
  programmable, maintainable etc.

Other goals:
  Allows other environments and machines to use the CM:
    VMS, Sun, MAC(?), Cray
  Allows other machines to use the CM without special hardware
    (using commercial networks)
  Supports low end CM

Philosophy of this proposal:
  We are building a kind of CM operating system.
    It has disks, and memory allocation, time sharing, user handling, metering, diagnostics, garbage collection, file system, network handling etc.
  Continue to buy what we can from the street and build what we can not buy.
  We can use a real computer to help us with the CM operating system.
    This I am calling the CM-Driver.
  We are driven by computation performance: There will be different system congurations (of disks, CM processors, cm-drivers, and front ends) based on the needs of different market segments. We must be flexible.
  Continue the "keep as much in software as performance allows" philosophy of the CM. This has allowed us to get where we are today.

Proposal pieces:
  1. Use UCC as "lls engine" and tightly coupled conventional box as a CM driver (we need a name for this).
  2. Macro-code: The UCC would execute Macro-code coming from the CM-driver.
  3. Macro-compiler that will run on the CM driver that can take expressions and procedures to produce macrocode for the machine.
  4. The CM Driver would handle memory management, user management (timesharing), and I/O for its CM.
  5. Front ends would then talk to this CM driver to run the CM code.

Features offered by this proposal:
  CM-1 compatibility
  No special front end hardware (hbi, bibi) (potential)
  Easy front end software interface
  Darpa net interface becomes easily defined
  Eliminate uVax in datavault (worth 20% in parts cost reduction) (potential)
  Direct way to incorporate numerical groups work in standard CM software
  Simplification of UCC
  Reduction of ucc control store by factor of 50.
  Increased CM performance for normal paris (over current levels).
  Increased CM performance for special optimized codes.
  Easy to code special optimized codes.
  Easy to add further "OS" features such as timesharing, I/O, Garbage collection, ...
  Dont need new version of assembler
  Coding paris is done in high level language
  Assemble-time scratch space allocation is eliminated (source of many ucode bugs).
  Stack space usage drops because of scoping VP looping outside of memory allocation.
  Eliminate Nexus (potential).
  1 machine type for the cm-driver.
  Small machine architecture defined.
  Large machine architecture defined.
  Many metering tools can be written in high level code (rather than microcode)

1. UCC as LLS engine:
  Use the UCC as an simple macrocode engine that can run a small number of macro-instructions with minimum overhead on the CM.
  Using the UCC as an LLS engine is less than it can do, thus we have all the hardware we need to make this happen. We could design a simpler board at some time to handle only the functions needed. The idea is to make this cheap, fast, and small.
  Currently there is a 1-3uSec Fifo push time, depending on this front end type and code optimization. With careful design, this could brought down to main memory cycle time (about 250Nsec). Since this only has to be done for one machine (the driver) and the physical distance is not great, this should not be too much trouble.

2. Macro-code:
  Currently the UCC runs VP paris on the CM. CAL and Dan are writing the new vp scheme in the driver and will run physical paris on the UCC and the rest stays in the driver. This is a good first step.
  A new interface (macrocode spec) can be derived that will give us the advantage of microcompiling while reducing the amount of microcode space and complexity. This interface will be derived by pulling together our microcoding experience.
  A macrocode set is something a compiler can put out (*lisp compiler or paris written in *x are two examples) that is run directly by microcode in a micro-engine (currently the ucc, but hopefully this board will get dramatically simpler, cheaper, and faster). Macrocode sets, these days, are typically a small number of powerful operations. We are somewhat limited by our instruction bandwidth from our host but 1-2 uSec/word aint so bad (and it could get better if we wanted it to). Macrocode, if well designed, makes "micro-compiling" unnecessary because one can get close to optimal preformance with using a good compiler. As we understand things like tag bits (context bit, general pvar bits) these will be incorportated, but until then we can support them in macrocode. Keep experiments out of microcode.
  The macrocode does not have to be pretty or orthogonal (alex and jeff mincy are the only ones that will have to deal with it alot). Further we could make a macrocode simulator that would replace the paris simulator.

Features (no implied order):
 maximize numeric performance codes (32 and 64 bit FPU float ops).
 maximize word-sized interger performance (16 and 32 bit).
 Small number of instructions.
 Minimize length for each instruction
   (dont make instructions too general that the common cases are too many bits to express)
 Minimize the number of instructions needed to express common sequences
   (minimize host overhead to make sure that the CM is busy).
 Minimize decoding in micro-engine (this is a RISK idea that keeps down the time to execute an instruction as well as keeps the complexity of the hardware down (control store).)
 Maximize nanoinstruction/microinstruction ratio in running real code.
   (minimize dispatch, decoding, and loop overhead)

Potential instruction set:

LLS-UP and some well chosen variants
LLS
PVO (of some sort)
  (Load-regfile, unload-regfile, load-transposer while loading-regfile...)
CALL
SET-REG
MOVE-REG
GET-REG
DISK-SEEK (and a few others for waiting for disk)
Dotimes (?)
if-global-call
Write-OFIFO
petit-cycle

3. Macro-compiler takes expressions and procedures and outputs object code that contains calls to macrocode. Currently this is the *lisp compiler and it puts out calls to VP paris. This proposal would extend the power of the compiler by allowing it more access to the internals of the LLS engine.
Many have observed (and demonstrated) that CM utilization can be enhanced if we get closer to the UCC and CM by microcoding than paris. If we design the macrocode set well then we can get this performance by putting smarts in the compiler.
This macro-compiler will eventually be used by all languages, but can start out in one (probably *lisp). Paris will continue to be supported by writing paris in macrocode. (one could imagine an optimizer that compiles strings of paris calls, but I am not sure if this is worthwhile.)

4. CM Driver would handle many operating system tasks. The CM driver is a general purpose machine that gives us good cost/performance and availablity of commercial networking cards. It is coupled closely with the UCC and with the user processes (both remote and local).
A piece of software that runs on this hardware is the CM driver (we need a better name).
There are two models for this software:
1. the commands from user processes are given to the driver which executes them on the CM directly. This would be the case if the program were running remotely and using a network to relay the commands.
2. for user processes that are running on the CM driver hardware this will not have the performance required. Running user processes on the driver is necessary for low end machines. The problem comes in on the process switch time. Mach, a new operating system, is supposed to handle this very well so the problem may go away on its own accord. Until then we can use shared pages and process locking to achieve the desired effects.

Tasks that this hardware does are:
VP looping
Time sharing
Memory allocation on CM
Processor allocation on CM
Garbage collection
I/O handling
Network interfaces to front ends
Metering
Talks to the data-vault front end (maybe the cm-driver could be the datavault front end (?))

Timesharing/GC has some issues to be worked out on maintainance of the field translation table, but I am confident we can work these out.

5. Front ends would talk to the CM-driver to execute code.
Conventional networks (ether, next-generation LANs, and WANs) will become the medium for this communication.
The data that is communicated over this medium will start at Paris and become more abstract. Eventually, expressions and procedures will be loaded into the driver and the front end will call these by name (RPC type). More work is needed here, but I hope the protocol will fall out as time goes on.
In the short term this interface might be virtual processor Paris calls or even Febi commands (see cal's implemented spec). This puts both throughput and latency requirements that rule out using the ether for performance critical applications. We might be able to use more conventional datapaths such as parallel ports or high speed networks (Potter?) rather than HBI's and Hbus's (yippie).

Conclusion:

With this strategy we can achieve the requirements and goals as well as make our machine much easier to program.
Hopefully this will:
make shipping the assembler unnecessary,
make the whitney-demo's switch unnecessary,
eliminate the nexus,

eliminate the HBI and BiBi,
simplify the UCC ($20K is too damn much),
allow timesharing (soon),
unify our languages at the instruction generation level,
decrease the number of microcoders in the company (increase the
  number of macrocoders),
increase CM performance.

REMOTE USE OF THE CM (CM-2 release 6):

```
         __________________________________________________________
        |       Front End                                          |
        |How many: Any number                                      |
        |Type: Anything (eg. 780 VAX, Lispm, MAC, BI VAX, Cray)    |
        |Operating system: any (VMS, unix, lispm, MAC, anything)   |
        |Memory Capacity:                                          |
        |       Infinite (it pages)                                |
        |Speed: Any                                                |
        |Functions:                                                |
        |       Familiar User environment, handles user,           |
        |       Executes user code, user time sharing, files,      |
        |       talks to CM drivers, serves error information,     |
        |       "London Interpreted" C*, *Lisp, Fortran.           |
        |__________________________________________________________|
                 |
                 |  network:      ether, DR11, hbus, anything commercial
                 |  protocol:     london, RPC-like
                 |  bandwidth:    1Mbit and up
                 |  latency:      2uSec to milisecs
         ________|________________________________
        |       Driver                            |
        |How many: 1 per CM (hopefully)           |
        |Type:  Sun-4 or comparable               |
        |       Characteristics:                  |
        |        Has low and high end             |
        |        very good $ per raw-mip ratio    |
        |        public bus                       |
        |        public OS (unix like)            |
        |        has every network interface      |
        |Speed: 10 MIPS or higher                 |
        |Memory Capacity:                         |
        |       Infinite (it pages)               |
        |Functions:                               |
        |       Network interface to FEs          |
        |       Local users running code          |
        |       Execute London Compiled code:     |
        |          C*,*lisp,Fortran(rpc)          |
        |       CM OS:                            |
        |        London Interpreter:              |
        |         Stack and heap management       |
        |         Field (pvar) type,              |
        |          lengths, sideways,             |
        |          in Weitek registers,           |
        |          paged in, etc.                 |
        |        London Compiler:                 |
        |         optimized paris generation      |
        |        Macrocompiler:                   |
        |         optimized macrocode             |
        |        VP looping,                      |
        |        Timesharing                      |
        |        Paging                           |
        |        File System (optional)           |
        |        Memory allocation                |
        |        Processor allocation             |
        |        Garbage collection               |
        |        Metering                         |
        |        Blink leds                       |
        |_________________________________________|
                 |
                 |  network:      hbus or something better (please)
                 |  protocol:     macrocode
                 |  bandwidth:    4MBytes/sec and up
                 |  Latency:      <200NSec
         ________|________________________________
```

```
|        Nexus                                  |
|How Many 1 per CM                              |
|Type: current design                           |
|Memory Capacity:                               |
|        32 bits                                |
|Function:                                      |
|        Clock                                  |
|        keeps multiple ucc's in synch          |
|        Handles multiple drivers               |
|         (hopefully not necessary)             |
-------------------------------------------------
          |
          | network:       gbus or something better (please)
          | protocol:      macrocode
          | bandwidth:     Same as hbus
          | Latency:       Same as hbus
          |
-------------------------------------------------
|        UCC                                    |
|How Many: 1 per 16K machine                    |
|Type: Current design                           |
|Speed: 8 MIPS                                  |
|Memory Capacity:                               |
|        64K control store (128bit words)       |
|        16K scratch RAM (32 bits words)        |
|Functions:                                     |
|        Executes Macrocode:                    |
|         lls-up                                |
|         WASM                                  |
|         PVO                                   |
|         DISK-SEEK                             |
|                                               |
-------------------------------------------------
          |
          | network:       backplane
          | protocol:      nanocode
          | bandwidth:     128Mbytes/sec
          | Latency:       5Nsec
          |
-------------------------------------------------
|        CM processors                          |
|How Many: lots                                 |
|Type: Home brew                                |
|Speed: high                                    |
|Memory Capacity:                               |
|        8Mbytes each                           |
|Functions:                                     |
|        Floating point (Weitek)                |
|        Integer and flags (Bit serial)         |
|        Routing (Router)                       |
|        Grid communication (router)            |
-------------------------------------------------
          |                                   |
          | network: backplane                |network: backplane
          | protocol:none to speak of         |protocol: none
          | bandwidth:125Mbytes/sec           |bandwidth: 125MBytes
          |                                   |
---------------------------------------  ------------------------------------
|        I/O Card                     |  |     Graphics board               |
|How Many: 1 per backplane            |  |How many:1 per backplane max      |
|Type: home brew                      |  |Type: home brew                   |
|Memory Capacity:                     |  |Memory Capacity: ?                |
|        16KBytes (buffer)            |  |Functions:                        |
|Functions:                           |  |     Look Awesome                 |
|        buffering I/O stream         |  |     Frame-grabbing(future)       |
|        Handle I/O bus               |  |                                  |
---------------------------------------  ------------------------------------
          |
          | network:       I/O Bus (home brew)
          | protocol:      I/O Bus Protocol (name?)
          | bandwidth:     80Mbytes/sec
```

```
        |
 ________________________________________________
|        Datavault                               |
|How Many: 1 per backplane max                   |
|Type: home brew                                 |
|Speed: 30Mbytes/sec                             |
|Memory Capacity:                                |
|        2Mbytes RAM (buffering)                 |
|        5-10 GBytes                             |
|Functions:                                      |
|        File storage                            |
|        Paging                                  |
|        Backups                                 |
|        Tape interface                          |
|________________________________________________|
```

CM Architecture today (CM-2, release 4)

```
 ______________________________________________________________
|        Front End                                             |
|How many: 4 max                                               |
|Type: BI VAX, lispm                                           |
|Operating system: ultix, lispm                                |
|Memory Capacity:                                              |
|        Infinite (it pages)                                   |
|Speed: Any                                                    |
|Functions:                                                    |
|        Familiar User environment, handles user,              |
|        Executes user code, files,                            |
|        serves error information,                             |
|        Interpreted *lisp, Compiled *lisp,                    |
|          *lisp:                                              |
|                 memory allocation,                            |
|                 stack and heap management                     |
|Compiled ~~"Interpreted~~" C*                                 |
|          C*:    memory allocation,                            |
|                 stack and heap management                     |
|        ~~VP Looping~~,                                        |
|        ~~Blink leds~~                                         |
|______________________________________________________________|
          |
          |  network:      hbus
          |  protocol:     macrocode
          |  bandwidth:    1-2MBytes/sec
          |  Latency:      <400NSec
          |
 ____________________________________________
|        Nexus                               |
|How Many 1 per CM                           |
|Type: current design                        |
|Memory Capacity:                            |
|        32 bits                             |
|Function:                                   |
|        Clock                               |
|        keeps multiple ucc's in synch       |
|        Handles multiple front ends         |
|____________________________________________|
          |
          |  network:      gbus or something better (please)
          |  protocol:     macrocode
          |  bandwidth:    Same as hbus
          |  Latency:      Same as hbus
          |
 ____________________________________________
|        UCC                                 |
|How Many: 1 per 16K machine                 |
|Type: Current design                        |
|Speed: 8 MIPS                               |
|Memory Capacity:                            |
|        64K control store (128bit words)    |
|        16K scratch RAM (32 bits words)     |
|Functions:                                  |
|        100's of paris instructions,        |
|        VP looping,                         |
|        blink leds                          |
|____________________________________________|
          |
          |  network:      backplane
          |  protocol:     nanocode
          |  bandwidth:    128Mbytes/sec
          |  Latency:      5Nsec
          |
 ____________________________________________
|        CM-1 processors                     |
|          Same as Remote              |
 _______________________________________
          |                        |
               .... same as remote ....
```

CM Architecture (CM-2 release 5 (march 88 release))

```
          ________________________________________________________________
         |        Front End                                               |
         |How many: Any number                                            |
         |Type: Anything (eg. 780 VAX, Lispm, MAC, BI VAX, Cray)          |
         |Operating system: any (VMS, unix, lispm, MAC, anything)         |
         |Memory Capacity:                                                |
         |        Infinite (it pages)                                     |
         |Speed: Any                                                      |
         |Functions:                                                      |
         |        Familiar User environment, handles user,                |
         |        Executes user code, user time sharing, files,           |
         |        talks to CM drivers, serves error information,          |
         |        "London Interpreted" *Lisp, Fortran(maybe).             |
         |                                                                |
          ________________________________________________________________
                  |
                  |  network:       ether, DR11, hbus, anything commercial
                  |  protocol:      london, RPC-like
                  |  bandwidth:     1Mbit and up
                  |  latency:       2uSec to milisecs
                  |
          _______________________________________________
         |        Driver                                 |
         |How many: 1 per CM (hopefully)                  |
         |Type:  Sun-4 or BI VAX or Lispm                 |
         |Speed: 1 MIP or higher                          |
         |Memory Capacity:                                |
         |        Infinite (it pages)                     |
         |Functions:                                      |
         |        Network interface to FEs(london)        |
         |        Local users running code                |
         |        Execute London Compiled code:           |
         |           *lisp, Fortran (maybe)               |
         |        Execute C* (without london)             |
         |        Fortran w/London Interpreter(maybe)|    |
         |        CM OS:                                  |
         |         London Interpreter:                    |
         |          Stack and heap management             |
         |          Field (pvar) type,                    |
         |           lengths, sideways,                   |
         |           in Weitek registers,                 |
         |           paged in, etc.                       |
         |         London Compiler:                       |
         |          optimized paris generation            |
         |         VP looping,                            |
         |         Timesharing                            |
         |         File System (optional)                 |
         |         Memory allocation                      |
         |         Blink leds                             |
         |                                                |
          _______________________________________________
                  |
                  |  network:       hbus or something better (please)
                  |  protocol:      macrocode
                  |  bandwidth:     4MBytes/sec and up
                  |  Latency:       <200NSec
                  |
          _______________________________________________
         |        Nexus                                   |
         |How Many 1 per CM                               |
         |Type: current design                            |
         |Memory Capacity:                                |
         |        32 bits                                 |
         |Function:                                       |
         |        Clock                                   |
         |        keeps multiple ucc's in synch           |
         |        Handles multiple drivers                |
         |         (hopefully not necessary)              |
 ----------------------------------------
          |
          | network:      gbus or something better (please)
          | protocol:     macrocode
          | bandwidth:    Same as hbus
          | Latency:      Same as hbus
          |
 ----------------------------------------
|       UCC                               |
|How Many: 1 per 16K machine              |
|Type: Current design                     |
|Speed: 8 MIPS                            |
|Memory Capacity:                         |
|       64K control store (128bit words)  |
|       16K scratch RAM (32 bits words)   |
|Functions:                               |
|       Executes Macrocode:               |
|        lis-up                           |
|        WASM                             |
|        PVO                              |
|        DISK-SEEK                        |
|        Diagnosics                       |
 ----------------------------------------
          |
          | network:      backplane
          | protocol:     nanocode
          | bandwidth:    128Mbytes/sec
          | Latency:      5Nsec
          |
 ----------------------------------------
|       CM processors                     |
|How Many: lots                           |
|Type: Home brew                          |
|Speed: high                              |
|Memory Capacity:                         |
|       8Mbytes each                      |
|Functions:                               |
|       Floating point (Weitek)           |
|       Integer and flags (Bit serial)    |
|       Routing (Router)                  |
|       Grid communication (router)       |
 ----------------------------------------
          |                               |
          | network: backplane            |network: backplane
          | protocol:none to speak of     |protocol: none
          | bandwidth:125Mbytes/sec       |bandwidth: 125MBytes
          |                               |
 ---------------------------------  ------------------------------
|       I/O Card                  | |      Graphics board          |
|How Many: 1 per backplane        | |How many:1 per backplane max  |
|Type: home brew                  | |Type: home brew               |
|Memory Capacity:                 | |Memory Capacity: ?            |
|       16KBytes (buffer)         | |Functions:                    |
|Functions:                       | |     Look Awesome             |
|       buffering I/O stream      | |     Frame-grabbing(future)   |
|       Handle I/O bus            | |                              |
 ---------------------------------  ------------------------------
          |
          | network:      I/O Bus (home brew)
          | protocol:     I/O Bus Protocol (name?)
          | bandwidth:    80Mbytes/sec
          |
 ----------------------------------------
|       Datavault                         |
|How Many: 1 per backplane max            |
|Type: home brew                          |
|Speed: 30Mbytes/sec                      |
|Memory Capacity:                         |
|       2Mbytes RAM (buffering)           |
|       5-10 GBytes                       |
|Functions:                               |
|         File storage                      |
|         Paging                            |
|         Backups                           |
|         Tape interface                    |
 -------------------------------------------
```